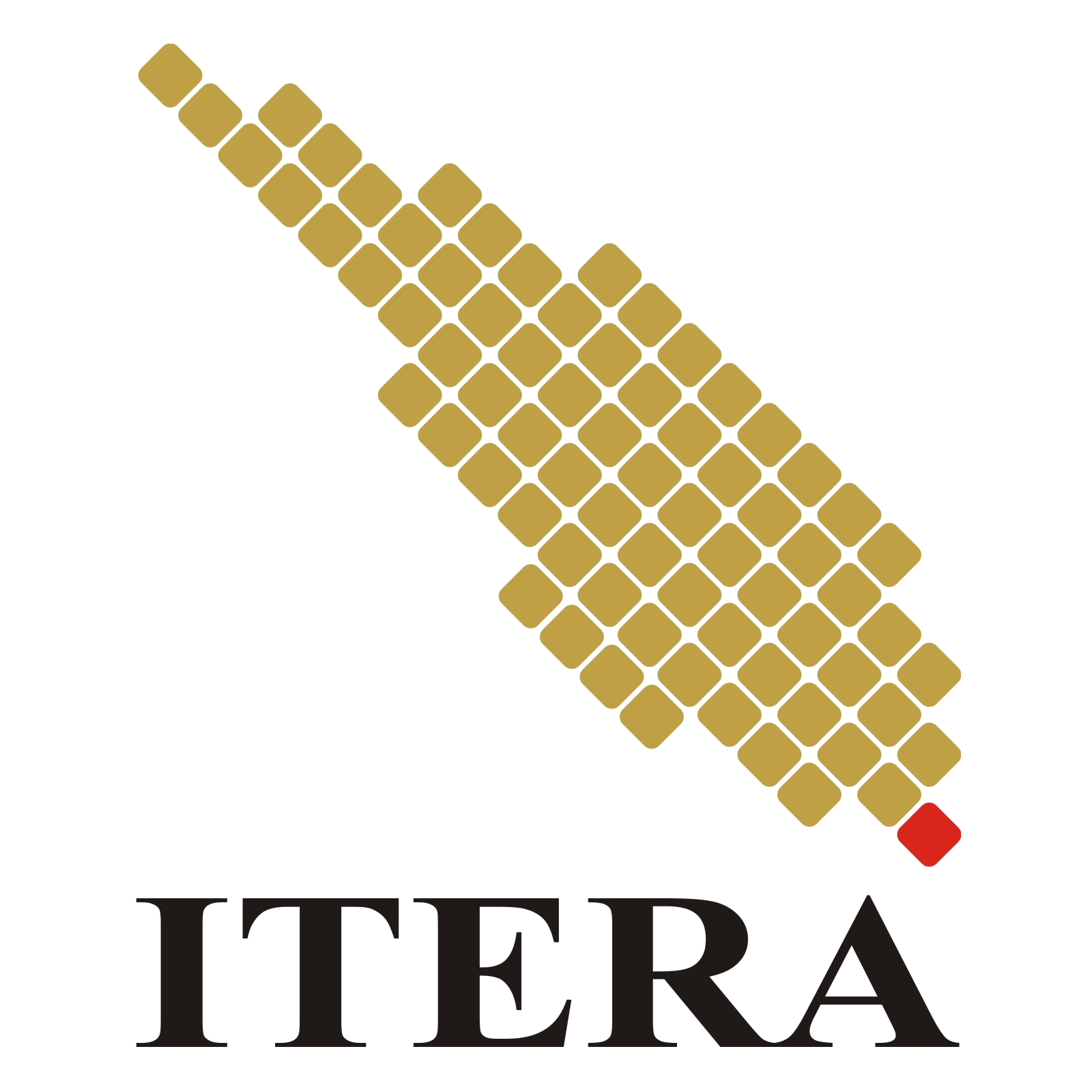
**LAPORAN TUGAS BESAR BASIS DATA  
PYTHON COURSES IN COURSERA**

KELOMPOK 7 - RA

****

**Dosen Pengampu**:

Riksa Meidy Karim, S.Kom., M.Si., M.Sc

Amalya Citra Pradana, S.Kom., M.Si., M.Sc.

Tirta Setiawan, S.Pd., M.Si..

**Anggota Kelompok :**

Dimas Wahyu Saputro (120450081)

Armanda Tantra Kumara (120450037)

Taj Shavira H. (120450059)

Fadia Dilla Sabine (120450011)

Khoirunnisa (120450029)

**INSTITUT TEKNOLOGI SUMATERA  
LAMPUNG SELATAN**

**2021/2022**

**1**

**Latar Belakang**

Saat ini banyak sekali bahasa pemrograman yang digunakan oleh para developer, salah satunya adalah Python. Karena maraknya dan peningkatan penggunaan bahasa pemrograman Python, tidak sedikit orang yang penasaran dan mempelajari bahasa pemrograman tersebut. Menurut survei yang dilakukan TIOBE Index, Python berada di peringkat ke-5 pada tahun 2016 sebagai bahasa pemrograman terpopuler. Python merupakan bahasa pemrograman tingkat tinggi karena dapat digunakan untuk membuat aplikasi baik berupa desktop, web serta mobile.

Oleh karena itu, pada tugas kali ini kami menggunakan dataset yang berisikan beberapa orang atau kelompok yang ikut dalam kursus pelatihan bahasa pemrograman Python. Data tersebut kami ambil dari [Python Courses on Coursera | Kaggle](https://www.kaggle.com/datasets/aayushsameershah/python-courses-on-coursera) dengan tujuan untuk memudahkan pengelolaan dari dataset tersebut menggunakan basis data. Coursera sendiri merupakan platform belajar online yang didirikan pada tahun 2012 oleh dua profesor asal Universitas Stanford.

Dalam dataset tersebut terdiri dari nama Universitas, course atau nama program yang diambil, tipe pemrograman, review atau penilaian atau rating terhadap course tersebut, jumlah peserta, pemilihan voting dan tingkat kesulitan dari course tersebut. Selanjutnya dataset tersebut diolah menggunakan sistem basis data atau DBMS *(Database Management System)*. Sistem basis data sendiri adalah sistem penyusunan dan pengelolaan rekaman-rekaman dengan menggunakan komputer, dengan tujuan untuk menyimpan atau merekam serta memelihara sata operasional lengkap dengan sebuah organisasi sehingga mampu menyajikan informasi yang diperlukan untuk pengambilan keputusan.

**Identifikasi Masalah**

Dari latar belakang diatas kita dapat merumuskan identifikasi masalah sebagai berikut :

1. Berapa course yang memiliki rating tertinggi dan terendah?
2. Berapa Universitas yang mengikuti course terbanyak?
3. Bagaimana tampilan baru setelah dikelola menggunakan sistem basis data?

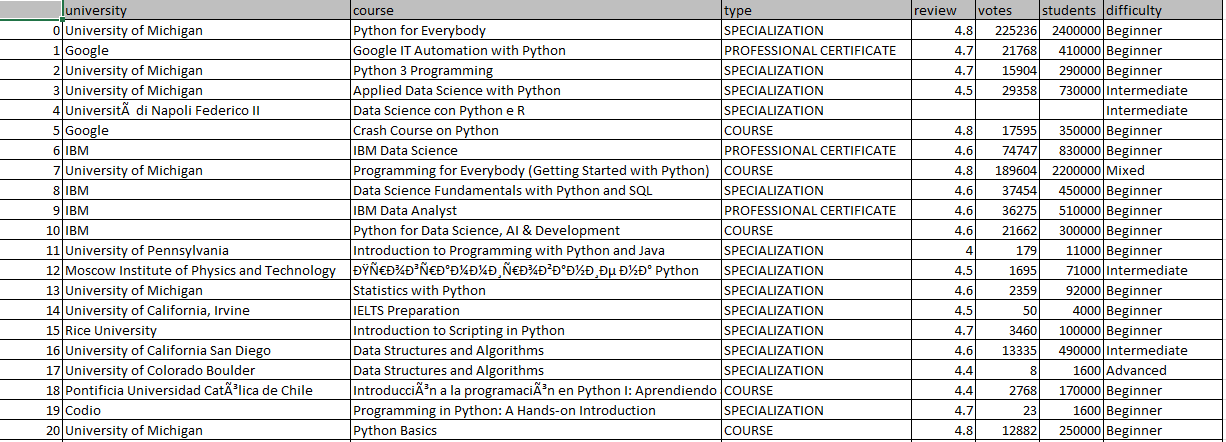
**Tujuan**

Adapun tujuan akhir dari tugas akhir ini adalah :

1. Memahami penggunaan sistem basis data dalam pengelolaan dataset.
2. Mengetahui jenis course dengan rating tertinggi dan terendah.
3. Mengetahui nama Universitas dengan mengikuti course terbanyak.

**2**

**Normalisasi pada Dataset**



Gambar 2.1 Data Awal

Relasi dengan data ganda menimbulkan banyak masalah. Selain boros ruang penyimpanan, ada tiga masalah utama yang muncul dan biasa disebut sebagai anomali update. Ketiga anomali update tersebut yaitu (1) anomali penambahan, (2) anomali penghapusan, dan (3) anomali modifikasi. Permasalahan anomali update ini dapat diatasi dengan normalisasi.

**Anomali yang bisa terjadi:**

Anomali merupakan proses pada basis data yang memberikan efek samping tidak diharapkan, anomali ini terjadi karena struktur tabel yang telah dibuat memiliki masalah. Masalah ini akan meningkatkan kompleksitas data dan menyulitkan dalam pemodifikasian data itu sendiri. Sehingga harus dilakukan cara-cara yang disebut normalisasi. Terdapat 3 macam anomali. Yaitu :

1. Anomali penambahan

Anomali penambahan dapat terjadi ketika ada data baru yang ditambahkan, tetapi ternyata terdapat elemen yang kosong dan elemen tersebut justru menjadi key.

1. Anomali penghapusan

Anomali penghapusan terjadi ketika suatu baris (tuple) yang tidak terpakai dihapus, kemudian sebagai akibatnya terdapat data lain yang hilang.

1. Anomali Modifikasi

Anomali modifikasi terjadi ketika ada sejumlah data yang diubah, tetapi tidak seluruhnya mengalami perubahan. Anomali modifikasi dapat menyebabkan ketidakkonsistenan pada data.

**Proses Normalisasi**

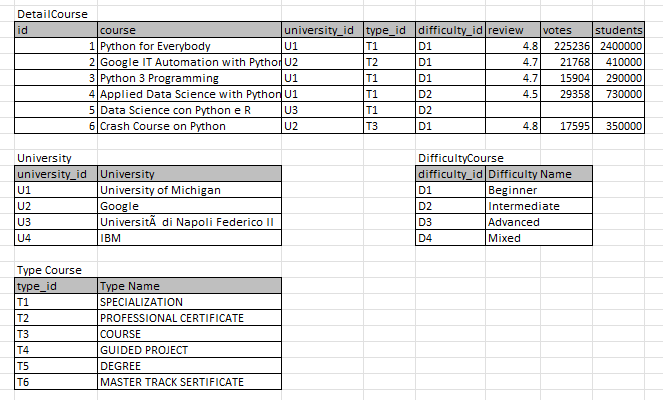
Normalisasi memiliki tujuan utama untuk mengelompokkan atribut ke dalam relasi sehingga data ganda menjadi minimal. Proses ini akan membagi tabel besar menjadi tabel yang lebih kecil dengan redudansi yang lebih sedikit. Tabel yang lebih kecil nantinya akan saling terkait satu sama lain melalui hubungan yang didefinisikan dengan baik.

1. 1NF (First Normal Form ‘Bentuk Normal Pertama’)  
   1NF merupakan bentuk normal pertama yang menyediakan sekumpulan persyaratan minimum untuk menormalkan database relasional. Suatu tabel dapat dikatakan 1NF jika dan hanya jika setiap atribut dari data tersebut hanya memiliki nilai tunggal dalam satu baris. Dalam 1NF, tidak diperbolehkan adanya *grouping* data atau duplikasi data.

Data yang digunakan telah memenuhi syarat 1NF karena data tersebut hanya memiliki nilai tunggal dalam satu baris.

1. 2NF (Second Normal Form ‘Bentuk Normal Kedua’)

2NF adalah bentuk normal kedua yang digunakan dalam database relasional. Agar tabel dapat memenuhi 2NF maka tabel tersebut harus dipatuhi dengan 1NF dan atribut apapun yang bukan merupakan bagian dari atribut e-prime harus sepenuhnya bergantung pada salah satu kandidat dalam tabel. Syarat 2NF adalah tidak diperkenankan adanya partial *functional dependency* kepada primary key dalam sebuah tabel.



Gambar 2. Sudah di normalisasi 2NF

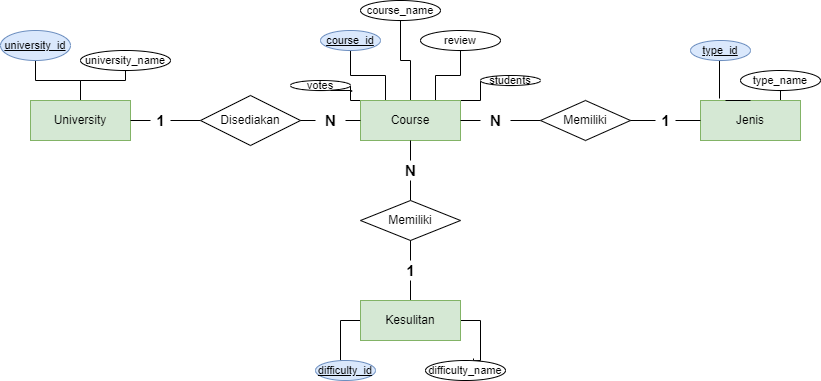
**3**

**Buat Relational Schema nya**

*Relation schema* (skema relasi) adalah nama relasi yang diikuti dengan kumpulan atribut-atributnya. Skema relasi merupakan sebuah cara yang dapat digunakan untuk merepresentasikan hubungan antara satu tabel dengan tabel lain melalui sebuah kolom kunci. Pada skema relasi, sebuah *primary key* suatu tabel merupakan *foreign key* pada tabel lainnya. Nantinya, kunci tersebut dapat digunakan untuk membantu kita menggabungkan informasi dari tabel-tabel yang terpisah.

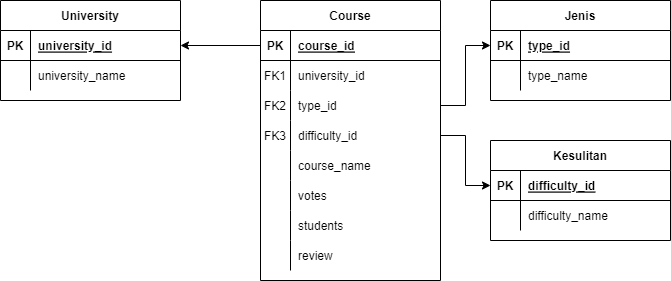
Kolom kunci merupakan suatu kolom yang berisi nilai-nilai yang menunjukkan sebuah observasi. Pada database relasional, terdapat dua jenis kolom kunci. Antara lain:

1. Primary Key: Primary key membentuk keunikan, yaitu hanya satu primary key yang diperbolehkan untuk setiap tabel. Primary key merupakan not null, atau dapat dikatakan juga sebagai constraint unique yang memiliki nilai not null.
2. Foreign Key: Digunakan sebagai hubungan antar primary atau unique ke yang lain. Foreign key bertujuan untuk menetapkan suatu kolom atau kombinasi dari beberapa kolom menjadi foreign key dari sebuah tabel. Foreign key disebut juga referential integrity constraint.



Gambar 3.1 ERD

* University mempunyai atribut : university\_id dan university\_name
* Course memiliki primary key course\_id dengan atribut votes, review, course\_name, dan student
* Jenis mempunyai primary key type\_id dengan atribut type\_name
* Kesulitan mempunyai primary key difficulty\_id dengan atribut difficulty\_name



Gambar 3.2 Relational Schema

Deskripsi permasalahan pada Gambar 3.2:

1. Course disediakan oleh university dan memiliki atribut: course\_id (primary key), university\_id (Foreign key 1), type\_id (Foreign Key 2), difficulty\_id (Foreign Key 3), course\_name, votes, students, dan review.
2. Course memiliki jenis : course type\_id berhubungan dengan jenis type\_id (primary key) dan memiliki atribut yaitu type\_name
3. Course memiliki tingkat kesulitan : dificulty\_id (primary key), difficulty\_name.

**4**

**Implementasi syntax - DDL (*Data Definition Language*)**

DDL merupakan singkatan dari D*ata Definition Language*, yaitu kumpulan perintah pada *Structured Query Language* (SQL) untuk menggambarkan desain dari database secara keseluruhan. Selain itu *Data Definition Language* (DDL) digunakan untuk membuat, menghapus, maupun merubah struktur definisi tipe data dari objek pada database. Dalam DDL terdapat perintah, yaitu CREATE, ALTER, DROP, RENAME, dan SHOW yang akan dieksekusi ketika diakhiri dengan ‘ ; ‘ (titik koma). Penulis disini menggunakan perintah CREATE.

Perintah ini dapat digunakan ketika membuat, termasuk dalam membuat database baru atau tabel baru. Penulis disini menggunakan sintaks ini untuk membuat database baru dan tabel baru.

1. CREATE DATABASE

Create database disini berguna untuk membuat database baru yaitu *tubes\_kel7\_final.*

CREATE DATABASE IF NOT EXISTS tubes\_kel7\_final;

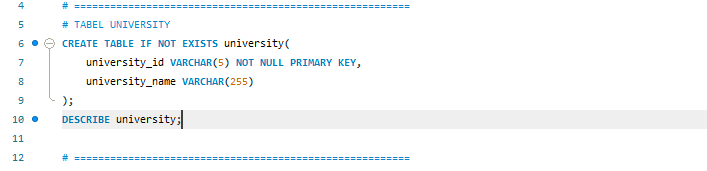
Setelah create database, database tersebut tidak dapat langsung digunakan. Untuk mengakses database yang telah dibuat menggunakan perintah

USE tubes\_kel7\_final;

1. CREATE TABLE

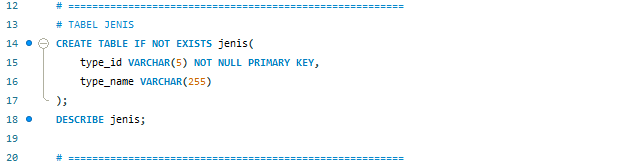
Create table disini berguna untuk membuat tabel baru pada database yang dibuat yaitu table, *university, jenis, kesulitan dan course.*

1. CREATE TABLE university

Gambar 4.1 Sintaks untuk membuat tabel university

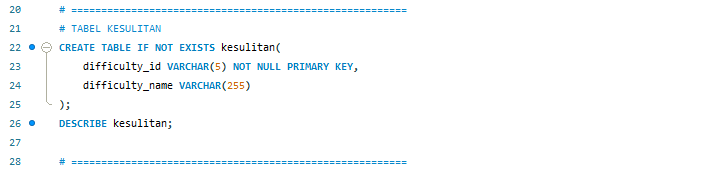
Dalam tabel terdapat CONSTRAINT untuk mendefinisikan, mengizinkan atau membatasi nilai apa yang dapat disimpan dalam kolom. Pada table university ini terdapat university\_id dengan CONSTRAINT yang digunakan yaitu PRIMARY KEY.

1. CREATE TABLE jenis

Gambar 4.2 Sintaks untuk membuat tabel jenis

Pada tabel jenis ini terdapat type\_id dengan CONSTRAINT yang digunakan yaitu PRIMARY KEY.

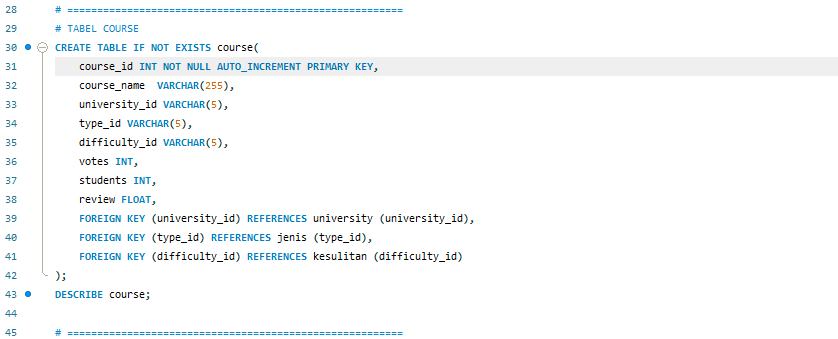
1. CREATE TABLE kesulitan



Gambar 4.3 Sintaks untuk membuat tabel kesulitan

Pada table kesulitan ini terdapat difficulty\_id dengan CONSTRAINT yang digunakan yaitu PRIMARY KEY.

1. CREATE TABLE course

Gambar 4.4 Sintaks untuk membuat tabel course

Pada table course ini terdapat difficulty\_id dengan CONSTRAINT yang digunakan yaitu PRIMARY KEY. FOREIGN KEY terdapat pada university\_id, type\_id, dan difficulty\_id.

**5**

**Implementasi Syntax - Data Manipulation Language**

DML atau Data Manipulation Language merupakan perintah SQL yang berhubungan dengan manipulasi atau pengolahan data dalam tabel. Ketika struktur sebuah database sudah terbentuk menggunakan perintah DDL atau Data Definition Language, langkah selanjutnya adalah mengolah data dalam tabel tersebut.

1. INSERT

Perintah ini untuk memasukkan sebuah record baru di dalam sebuah tabel database. Penulis menggunakan sintaks ini untuk memasukkan data ke dalam tabel *university*, tabel *jenis*, tabel *kesulitan*, dan tabel *course*.



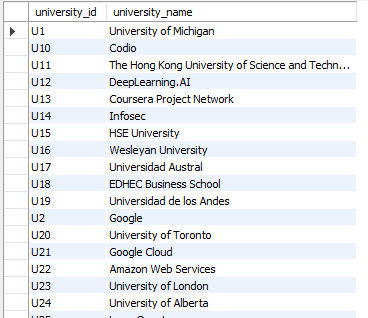
Gambar 5.1 Sintaks untuk memasukkan data ke tabel university

Sintaks Insert penulis gunakan untuk tabel yang lainnya, seperti tabel *jenis*, tabel *kesulitan*, dan tabel *course.*

1. SELECT

Select adalah sintaks yang digunakan untuk mengambil data dari objek database seperti tabel. Untuk mengambil semua baris data dapat menggunakan \* . Jika data yang diambil spesifik maka dapat menggunakan nama kolom. Nama kolom yang dipilih dapat berjumlah lebih dari satu. Untuk mendapatkan data yang lebih spesifik, dapat menggunakan where untuk mendefinisikan kondisi.

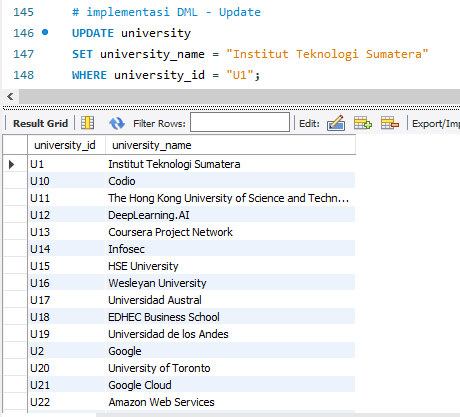
Pada tugas akhir ini, penulis menggunakan sintaks *Select* untuk memeriksa apakah data yang dimasukkan saat melakukan *Insert* sudah benar atau belum. Ketika sintaks pada Gambar 5.1 dijalankan, akan mengisi kolom *university\_id*, dan *university\_name*. Hasilnya terlihat seperti pada Gambar 5.2.



Gambar 5.2 Syntax Select

1. UPDATE

Update adalah sintaks yang memungkinkan untuk memperbarui data tabel yang ada. Untuk mengubah baris data yang lebih spesifik dapat menggunakan where untuk mendefinisikan kondisi. Nama kolom yang akan diubah dapat berjumlah lebih dari satu. Misalnya, penulis ingin mengubah *university\_name* menggunakan *university\_id = U1.* Maka, yang dilakukan terlihat seperti pada Gambar 5.3. Telihat bahwa *university\_name* pada *university\_id* = U1 sudah berubah menjadi Institut Teknologi Sumatera.



Gambar 5.3 Sintaks Update

Terlihat pada Gambar 5.3 bahwa *university\_name* pada *university\_id* = U1 sudah berubah menjadi Institut Teknologi Sumatera.

1. DELETE

Delete adalah sintaks yang dapat menghapus baris data. Sintaks delete fungsinya mirip dengan sintaks truncate. Perbedaan sintaks delete adalah dapat menentukan dengan tepat apa yang ingin dihapus, sedangkan truncate memungkinkan untuk menghapus semua record yang terdapat dalam sebuah tabel. Untuk menghapus baris data yang lebih spesifik dapat menggunakan where untuk mendefinisikan kondisi.

Karena data pada tabel terlihat baik-baik saja, maka penulis tidak menggunakan sintaks ini. Namun, jika ingin menggunakan sintaks ini, caranya sebagai berikut:

mysql> **TRUNCATE TABLE** nama\_tabel ;

atau

mysql> **DELETE FROM** nama\_tabel ;

atau

mysql> **DELETE FROM** nama\_tabel

WHERE nama\_kolom = kondisi\_nilai\_dicari

;

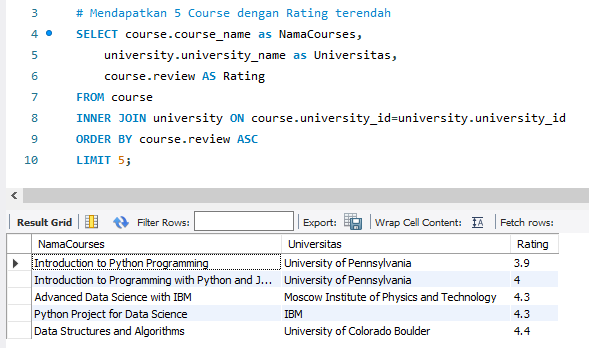
**6**

**Advanced Query - Insight**

DML atau Data Manipulation Language merupakan perintah SQL yang berhubungan dengan manipulasi atau pengolahan data dalam tabel. Ketika struktur sebuah database sudah terbentuk menggunakan perintah DDL atau Da

1. Mendapatkan 5 course dengan rating terendah.

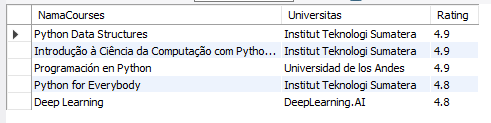
Untuk mendapatkan course dengan rating terendah, diperlukan kolom *review* pada tabel *course*. Jika hanya mengambil kolom *review*, maka akan kurang berguna. Oleh karena itu, penulis menambahkan kolom *course\_name* dari tabel *course,* dan *university\_name* dari tabel *university*. Selanjutnya, karena berasal dari tabel yang berbeda, maka diperlukan *join*. Setelah dilakukan *join,* dilakukan pengurutan menggunakan *order by.* Hasil yang didapatkan terlihat pada Gambar 6.1 yaitu Course *Introduction to Python Programming* oleh *University of Pennslylavania* dengan rating 3.9 menempati peringkat pertama.



Gambar 6.1 Lima Course dengan Rating Terendah

1. Mendapatkan 5 course dengan rating terbaik.

Cara yang dilakukan sama seperti sebelumnya. Hanya saja terdapat perbedaan pada pengurutan menggunakan *Order By.* Kali ini digunakan DESC, sehingga hasil yang didapatkan menurun. Hasilnya terlihat pada gambar 6.2.



Gambar 6.2 Lima Course dengan Rating Terbaik

1. Mendapatkan rata-rata rating dari setiap universitas pembuat

Rata-rata rating dari setiap universitas yang mengeluarkan course sangat menarik untuk dilihat. Namun, jika yang dilihat hanya rating saja, data yang didapatkan akan kurang bagus. Penulis disini mengikutsertakan jumlah course dari setiap universitas. Caranya menggunakan sintaks di bawah ini:

# Mendapatkan rata-rata rating dari setiap universitas

SELECT university.university\_name as Universitas,

COUNT(course.review) AS JumlahCourse,

AVG(course.review) AS RatingAVG

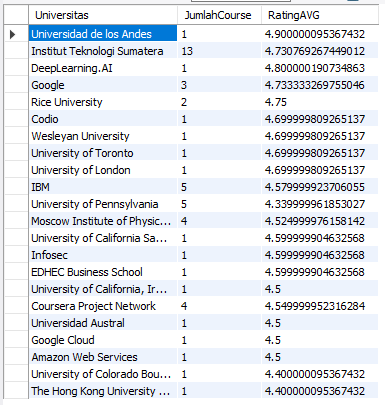
FROM course

INNER JOIN university ON course.university\_id=university.university\_id

GROUP BY course.university\_id

ORDER BY course.review DESC;

Ketika sintaks di atas dijalankan, hasilnya akan terlihat seperti pada Gambar 6.3. Hal yang menarik disini, adalah Institut Teknologi Sumatera meskipun memiliki banyak course yang banyak, namun rata-rata ratingnya tetap tinggi.



Gambar 6.3 Rata-rata rating berdasarkan Universitas Pembuat

1. Mendapatkan Universitas dengan Jumlah Course Lebih dari Tiga

Dari insight nomor tiga, sangat menarik untuk mengetahui universitas apa yang memiliki jumlah course lebih dari tiga. Hal itu dapat dilakukan dengan query di bawah.

# Mendapatkan universitas dengan jumlah course lebih dari tiga

SELECT university.university\_name as Universitas,

COUNT(course.course\_id) AS JumlahCourse

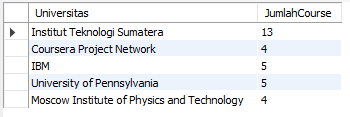
FROM course

INNER JOIN university ON course.university\_id=university.university\_id

GROUP BY course.university\_id

HAVING COUNT(course.course\_id) > 3;

Setelah query di atas dijalankan, didapatkan hasil seperti pada Gambar 6.4. Hal yang menarik, dari 30 universitas, ternyata hanya 5 universitas yang memiliki course lebih dari tiga.



Gambar 6.4 Universitas yang Memiliki Course Lebih dari Tiga

1. Menggabungkan seluruh tabel yang ada, dan hanya menampilkan kolom yang penting

Karena penulis memiliki *super computer,* penulis tertarik untuk menggabungkan seluruh tabel yang ada menggunakan *join*. Query yang digunakan sebagai berikut:

# Menggabungkan seluruh tabel, tapi hanya menampilkan kolom yang penting

SELECT course\_id, course\_name, university\_name, type\_name, difficulty\_name, votes, students, review

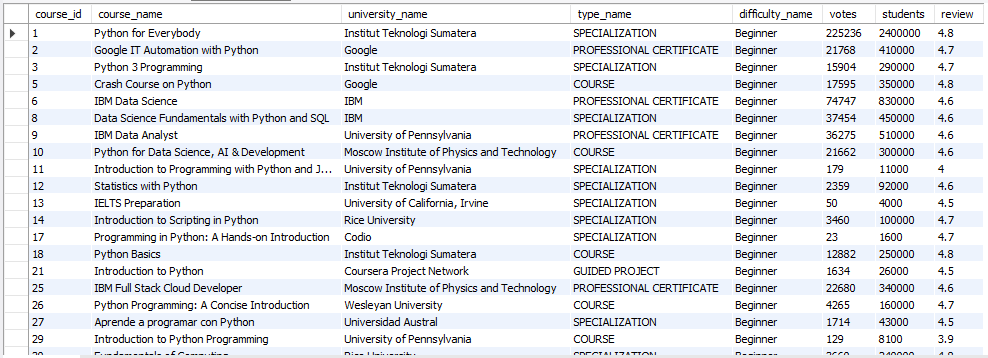
FROM course

INNER JOIN university ON course.university\_id=university.university\_id

INNER JOIN jenis ON course.type\_id=jenis.type\_id

INNER JOIN kesulitan ON course.difficulty\_id=kesulitan.difficulty\_id;

Ketika Query di atas berhasil dijalankan, akan menghasilkan output seperti pada gambar 6.5. Terlihat perubahan yang menarik dari tabel. Pada awalnya, tabel hanya terlihat foreign key saja. Namun, ketika dilakukan join, akan terlihat bagaimana data sebenarnya.



Gambar 6.5 Join Seluruh Tabel